Mata Kuliah : Sistem Operasi

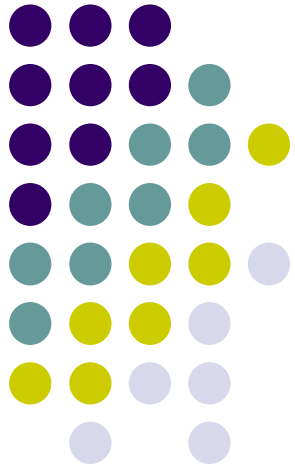
Kode MK : IT-012336

6

# Penjadualan CPU

Tim Teaching Grant

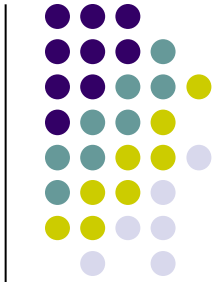
Mata Kuliah Sistem Operasi



# Penjadualan CPU

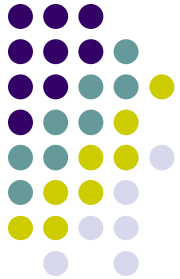
- Konsep Dasar
- Kriteria Penjadualan
- Algoritma Penjadualan
- Penjadualan Multiple-Processor
- Penjadualan Real-Time
- Evaluasi Algorithm

# Konsep Dasar



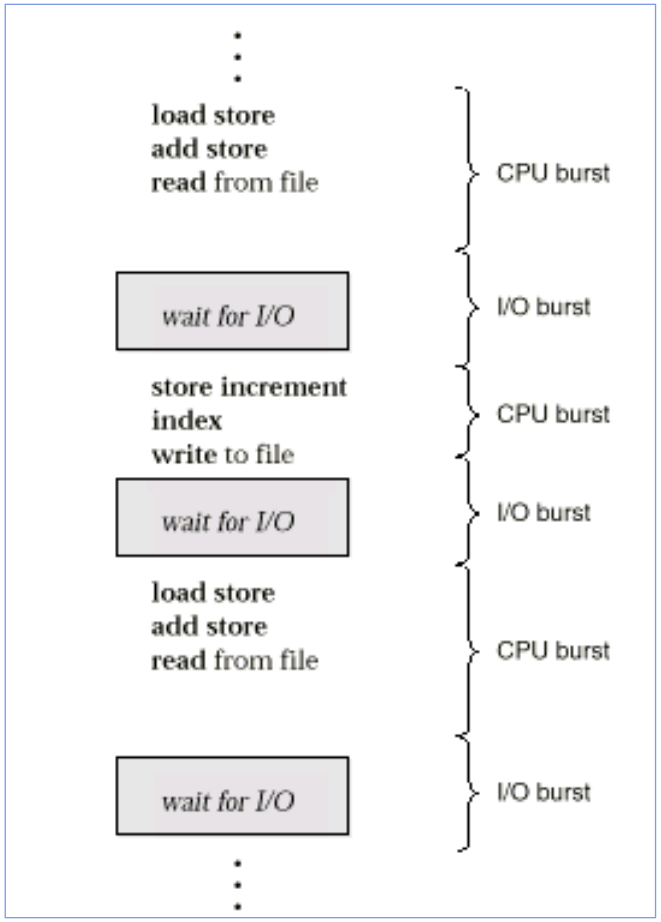
- Memaksimalkan kinerja CPU melalui multiprogramming
- CPU–I/O Burst Cycle
  - Eksekusi proses terdiri dari siklus eksekusi CPU dan I/O wait.
- Pendistribusian CPU burst

# Siklus CPU-I/O Burst (1)

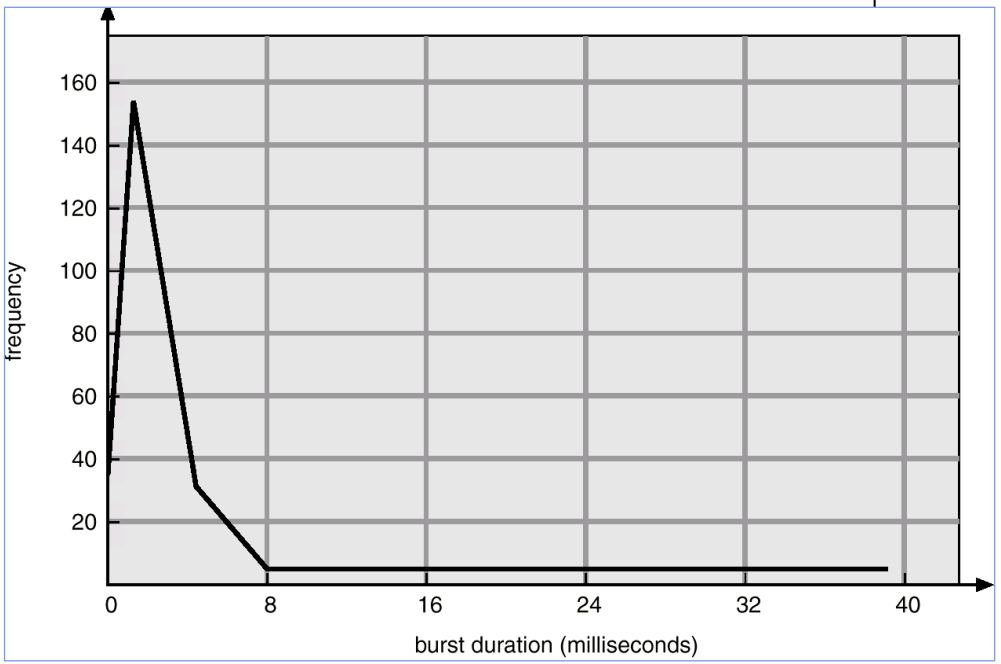


- Penjadwalan CPU tergantung properti proses sbb:
  - Eksekusi proses terdiri dari siklus eksekusi CPU dan menunggu I/O
  - Proses berpindah antara kedua state tersebut
  - Eksekusi proses dimulai dengan CPU burst, diikuti I/O burst, kemudian diikuti dengan CPU burst lain, I/O burst lain dan seterusnya
  - CPU burst terakhir akan berakhir dengan sistem meminta terminasi eksekusi bukan karena I/O burst yang lain
- Program I/O bound mempunyai banyak CPU burst yang sangat pendek
- Program CPU bound mempunyai beberapa CPU burst yang sangat panjang

# Siklus CPU-I/O Burst (2)




Urutan CPU dan I/O Burst

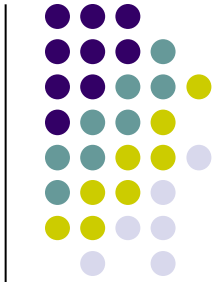



Histogram waktu CPU Burst

# Penjadualan CPU

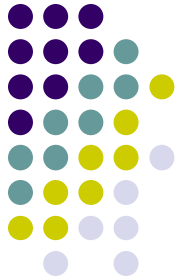
- Keputusan penjadwalan CPU mempertimbangkan 4 keadaan sbb :
  1. Ketika proses berpindah dari state running ke state waiting (contoh: permintaan I/O, atau menunggu terminasi dari satu proses child)
  2. Ketika proses berpindah dari state running ke state ready (contoh: saat terjadi interrupt)
  3. Ketika proses berpindah dari state waiting ke state ready (contoh: menyelesaikan I/O)
  4. Ketika proses diterminasi
- Penjadualan 1 dan 4 termasuk nonpreemptive
- Penjadualan lainnya termasuk preemptive

# Jenis Penjadualan



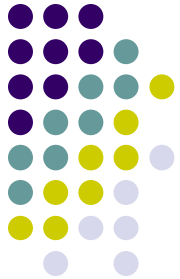
- Non-preemptive: setiap proses secara sukarela (berkala) memberikan CPU ke OS.
- Preemptive: OS dapat mengambil (secara interrupt, preempt) CPU dari satu proses setiap saat.
  - *Prasyarat untuk OS real-time system*

# Dispatcher



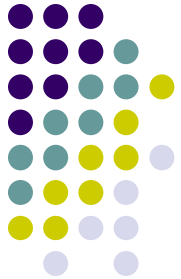
- *Dispatcher* adalah modul yang memberikan kontrol pada CPU terhadap proses yang dipilih dengan short-term scheduling.
- Fungsi-fungsinya :
  - Switching context
  - Switching ke user-mode
  - Melompat ke lokasi tertentu pada user program untuk memulai program
- Karena dispatcher digunakan setiap berpindah proses, dispatcher harus secepat mungkin
- Waktu yang dibutuhkan dispatcher untuk menghentikan suatu proses dan memulai menjalankan proses yang lain disebut *dispatch latency*
  - Save (proses lama) dan restrore (proses baru).

# Kriteria Penjadualan



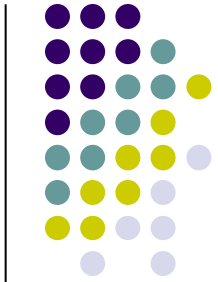
- Utilisasi CPU
  - menjadikan CPU terus menerus sibuk (menggunakan CPU semaksimal mungkin).
- Throughput
  - jumlah proses yang selesai dijalankan (per satuan waktu).
- Turn around time
  - waktu selesai eksekusi suatu proses (sejak di submit sampai selesai).
- Waiting time
  - waktu tunggu proses (jumlah waktu yang dihabiskan menunggu di ready queue).
- Response time
  - waktu response dari sistim terhadap user (interaktif, time-sharing system), sehingga interaksi dapat berlangsung dengan cepat.

# Kriteria Penjadualan yang Optimal



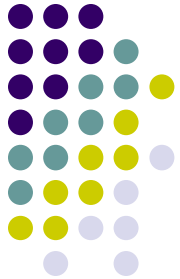
- Memaksimumkan utilisasi CPU
- Memaksimumkan throughput
- Meminimukan turnaround time
- Meminimumkan waiting time
- Meminimumkan  response time

# Algoritma Penjadualan



- Penjadwalan CPUadalahpermasalahan menentukan proses mana pada readyqueue yangdialokasikan ke CPU
- Terdapat beberapa algoritma penjadwalan CPU
  - First-come, first-served (FCFS)
  - Shortest-Job-First (SJF)
  - Priority Scheduling
  - Round-Robin (RR)
  - Multilevel Queue
  - Multilevel Feedback Queue

# First Come First Served (FCFS)



- Algoritma:
  - Proses yang request CPU pertama kali akan mendapatkan jatah CPU.
  - Sederhana – algoritma maupun struktur data: menggunakan FIFO queue (ready queue).
- Termasuk non preemptive
  - Timbul masalah “waiting time” terlalu lama jikadidahului oleh proses yang waktu selesainya lama.
    - Tidak cocok untuk time-sharing systems.
    - Digunakan pada OS dengan orientasi batch job.

# Contoh 1

| Proses | Burst Time |
|---|---|
| $P_1$ | 24 |
| $P_2$ | 3 |
| $P_3$ | 3 |

- Diketahui proses yang tiba adalah $P_1$, $P_2$, $P_3$. Gant chart-nya adalah :

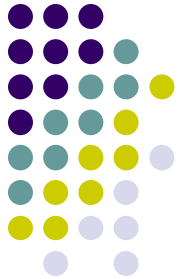

- Waiting time untuk *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

# Contoh 2

- Diketahui proses yang tiba adalah $P_2$, $P_3$, $P_1$. Gant chart-nya adalah :


- Waiting time untuk *P1* = 6*; P2* = 0*; P3* = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
  - Lebih baik dari kasus sebelumnya
- *Convoy effect* proses yang panjang diikuti proses yang pendek

# Shortest-Job-First (SJF)

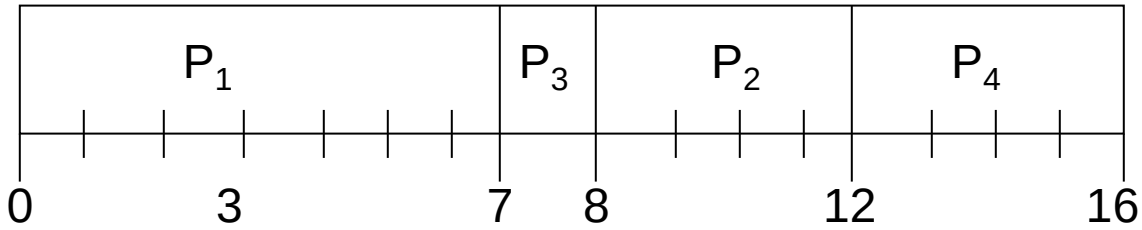


- Proses yang memiliki CPU burst paling kecil dilayani terlebih dahulu.
- Terdapat 2 skema :
  - *nonpreemptive* – CPU hanya satu kali diberikan pada suatu proses, maka proses tersebut tetap akan memakai CPU hingga proses tersebut melepaskannya
  - *preemptive* –jika sisa waktu proses pertama lebih besar dari proses kedua, maka proses pertama dihentikan dan diganti proses kedua. (dikenal dengan Shortest-Remaining-Time-First /SRTF).
- SJF akan optimal, ketika rata-rata waktu tunggu minimum untuk set proses yang diberikan

# Contoh Non-Preemptive SJF

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0.0 | 7 |
| $P_2$ | 2.0 | 4 |
| $P_3$ | 4.0 | 1 |
| $P_4$ | 5.0 | 4 |

- SJF (non-preemptive)

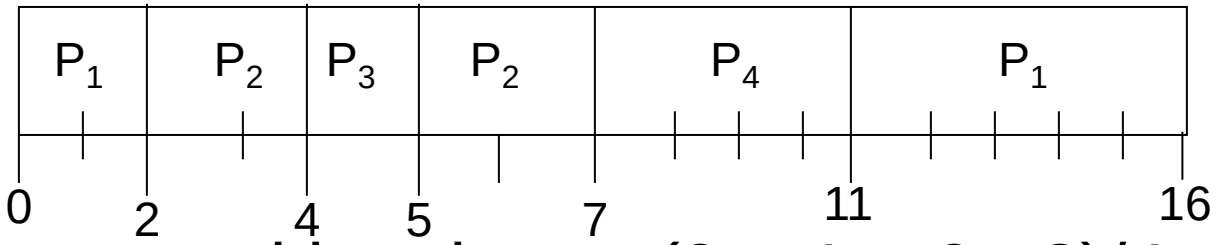



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

# Contoh Preemptive SJF

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0.0 | 7 |
| $P_2$ | 2.0 | 4 |
| $P_3$ | 4.0 | 1 |
| $P_4$ | 5.0 | 4 |

- SJF (preemptive)




- Average waiting time = (9 + 1 + 0 +2)/4 = 3

# Contoh lain

| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 6 |
| P2 | 2 | 8 |
| P3 | 3 | 7 |
| P4 | 5 | 3 |

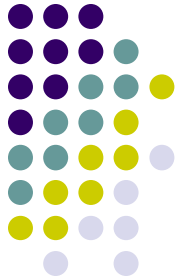
**Shortest Job First**

| Proses | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0 | 8 |
| $P_2$ | 1 | 4 |
| $P_3$ | 2 | 9 |
| $P_4$ | 3 | 5 |



**Shortest remaining time first**

# Penjadualan Prioritas

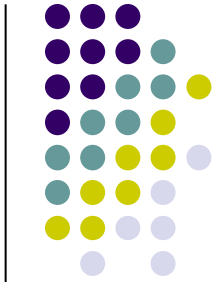


- Setiap proses dilengkapi dengan prioritas (bilangan integer).
- CPU diberikan ke proses dengan prioritas tertinggi (smallest integer highest priority)
- Jika prioritas sama, digunakan algoritma FCFS.
- Prioritas menyangkut masalah waktu,  memori, banyaknya file yang boleh dibuka dan perbandingan rata-rata I/O burst dengan CPU burst.
- Bersifat preemptive dan non preemptive
  - Preemptive : proses dapat di interupsi jika terdapat prioritas lebih tinggi yang memerlukan CPU.
  - Non preemptive : proses dengan prioritas tinggi akan mengganti pada saat pemakaian time-slice habis.

# Contoh

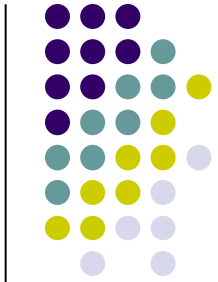
| Proses | Arrival | Burst | Priority |
|---|---|---|---|
| P1 | 0 | 10 | 3 |
| P2 | 2 | 1 | 1 |
| P3 | 3 | 2 | 3 |
| P4 | 5 | 1 | 4 |
| P5 | 7 | 5 | 2 |

# Problem : Starvation



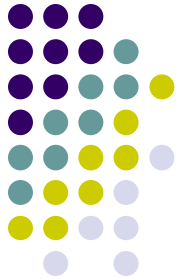
- Proses dengan prioritas terendah mungkin tidak akan pernah dieksekusi
- Solution : *Aging*, prioritas akan naik jika proses makin lama menunggu waktu jatah CPU

# Round Robin (1)



- Konsep dasar : *time sharing*
- Sama dengan FCFS yang bersifat preemptive
- Quantum time untuk membatasi waktu proses
- Jika CPU burst < *Quantum time*, proses melepaskan CPU jika selesai dan CPU digunakan untuk proses selanjutnya
- Jika CPU burst > *Quantum time*, proses dihentikan sementara dan mengantri di ekor dari *ready queue*, CPU menjalankan proses berikutnya

# Round Robin (2)

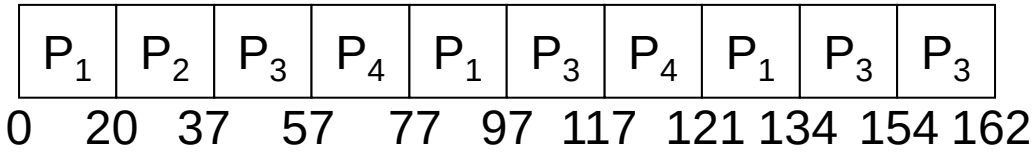


- Setiap proses mendapat jatah waktu CPU (time slice/quantum) tertentu misalkan 10 atau 100 milidetik.
  - Setelah waktu tersebut maka proses akan di-preempt dan dipindahkan ke ready queue.
  - Adil dan sederhana.
- Jika terdapat n proses di “ready queue” dan waktu quantum q (milidetik), maka:
  - Maka setiap proses akan mendapatkan 1/n dari waktu CPU.
  - Proses tidak akan menunggu lebih lama dari: (n-1) q time units.
- Performance
  - *q* besar $\Rightarrow$ FIFO
  - *q* kecil $\Rightarrow$ *q* harus lebih besar dengan mengacu pada context switch, jika tidak overhead akan terlalu besar

# Contoh RR (Q= 20)

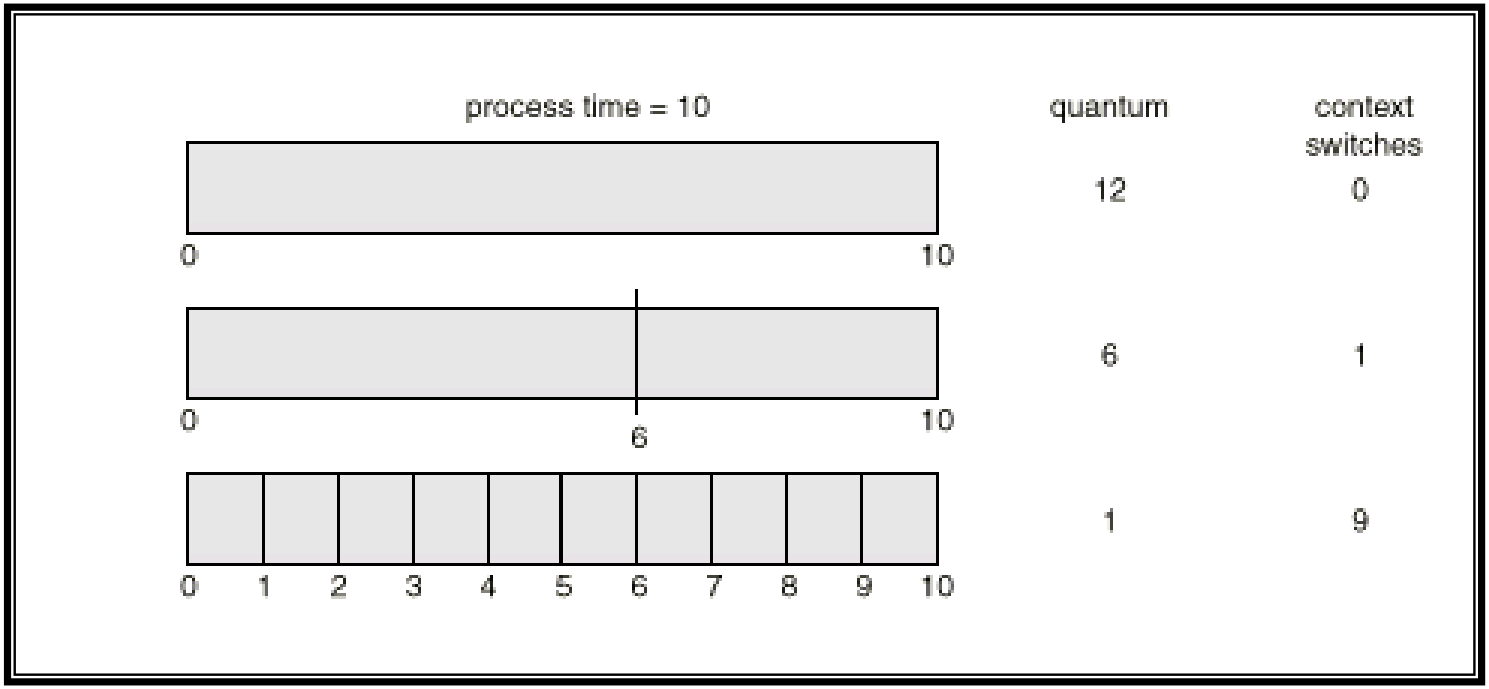
| Process | Burst Time |
|---|---|
| $P_1$ | 53 |
| $P_2$ | 17 |
| $P_3$ | 68 |
| $P_4$ | 24 |

- Gantt Chart

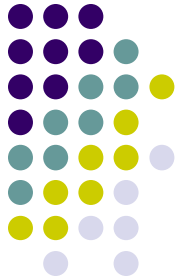



- Tipikal: lebih lama waktu rata-rata turnaround dibandingkan SJF, tapi mempunyai response terhadap user lebih cepat.

# Waktu Kuantum dan Waktu Context Switch

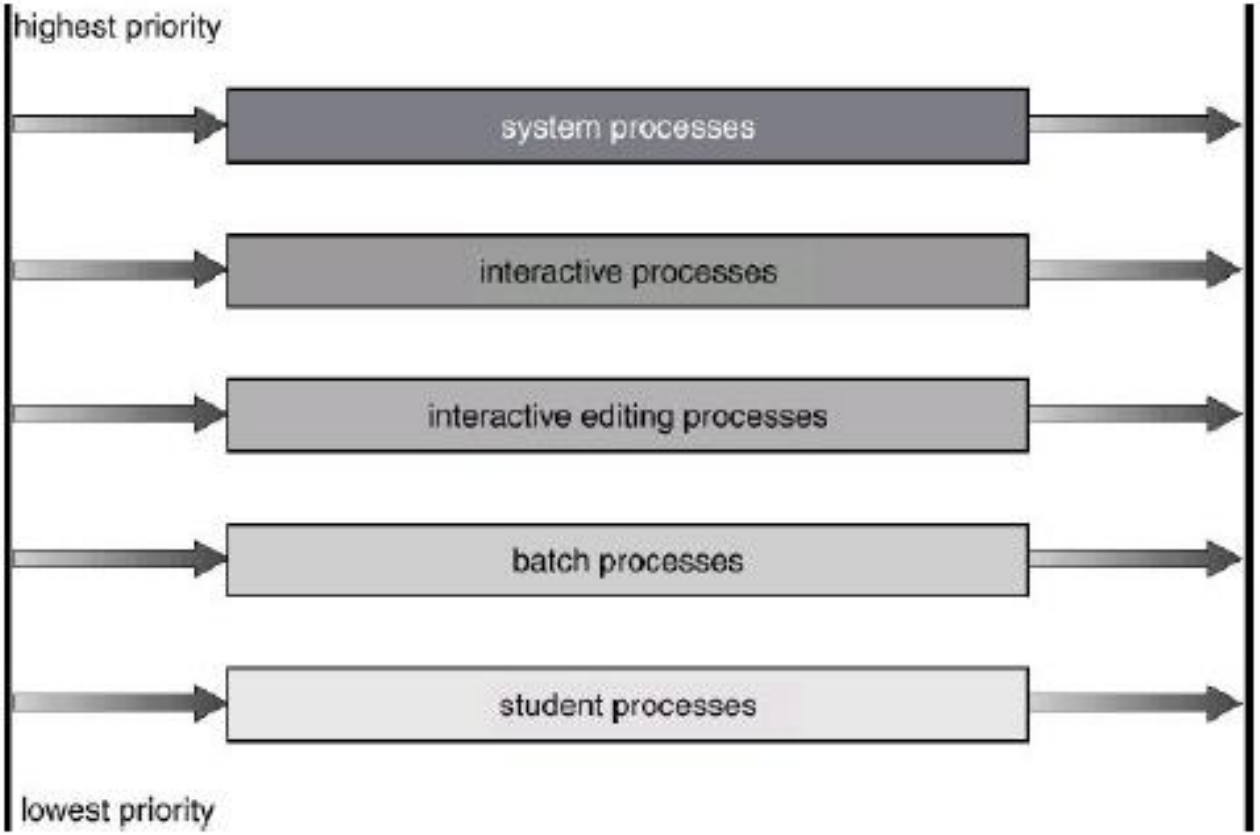


# Penjadualan Antrian Multitingkat

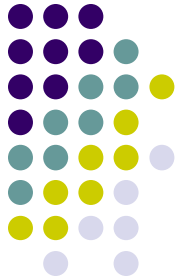


- Kategori proses sesuai dengan sifat proses:
  - Interaktif (response cepat)
  - Batch dll
- Partisi “ready queue” dalam beberapa tingkat (multilevel) sesuai dengan proses:
  - Setiap queue menggunakan algoritma schedule sendiri
  - Foreground proses (interaktif, high prioritiy): RR
  - Background proses (batch, low priority): FCFS
- Setiap queue mempunyai prioritas yang fixed.

# Penjadualan Antrian Multitingkat

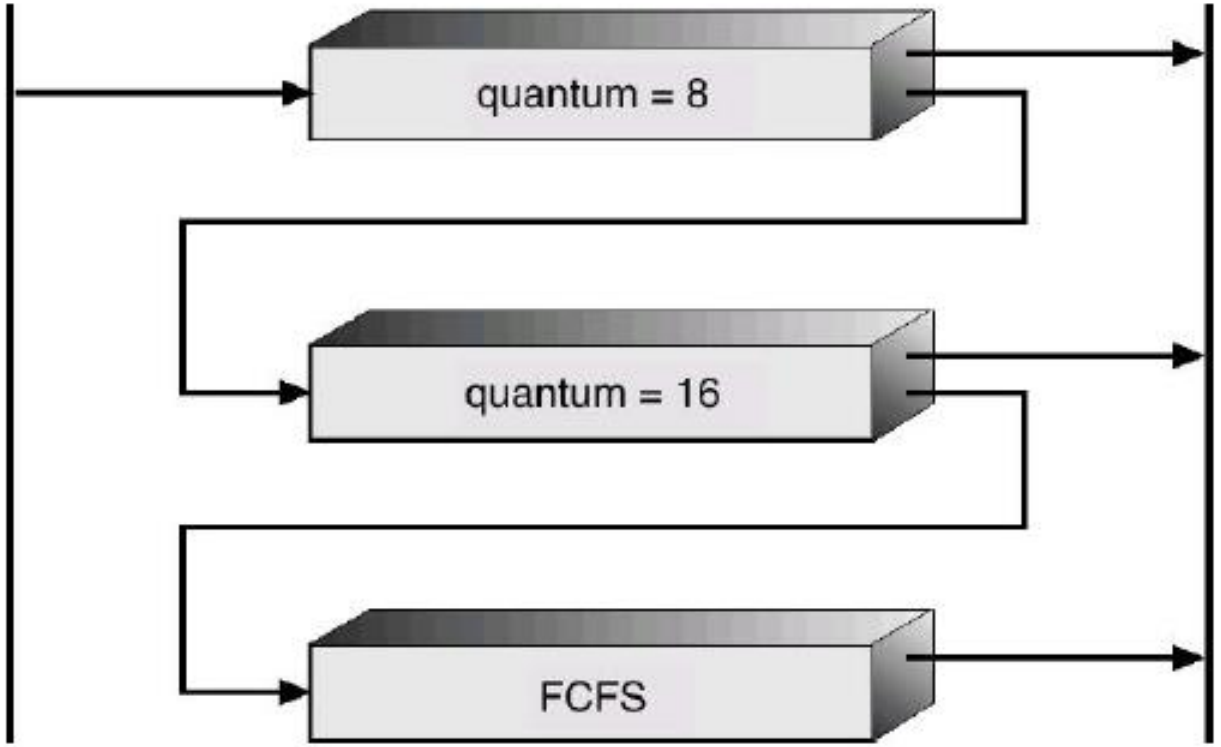


# Antrian Multitingkat Berbalikan

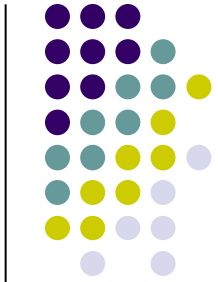


- Suatu proses dapat berpindah diantara beragam antrian;
- Perlu feedback untuk penentuan proses naik/turun prioritasnya (dinamis):
  - Aging dapat diimplementasikan sesuai dengan lama proses pada satu queue.
  - Suatu proses yang menggunakan CPU sampai habis (tanpa I/O wait) => CPU-bound (bukan proses interaktif) dapat dipindahk ke queue dengan prioritas lebih rendah

# Antrian Multitingkat Berbalikan



# Penjadualan Multiple-Processor

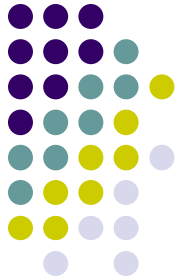


- Penjadualan CPU lebih kompleks ketika terdapat multiple Processor
- Processor yang homogen termasuk ke dalam multiprocessor
- *Load sharing*
- *Asymmetric multiprocessing* – hanya ada satu processor yang dapat mengakses struktur sistem data,sehingga meringankan kebutuhan sharing data

# Penjadualan Real-Time

- *Hard real-time* systems
  - Task kritis harus selesai dengan garansi waktu tertentu
  - OS akan melacak lamanya task tersebut dieksekusi (real time):
    - Mengetahui lama waktu system call, fungsi dan response dari hardware
    - Melakukan prediksi apakah task tersebut dapat dijalankan.
  - Mudah dilakukan untuk OS khusus pada peralatan/ pemakaian khusus (single task: control system)
  - Sulit untuk time-sharing sistim, virtual memory (faktor latency sebagian program aktif ada di disk).

# Penjadualan Real-Time



- *Soft real-time* systems
  - Membutuhkan penggunaan skema prioritas
  - Multimedia, highly interactive graphics
  - Prioritas tidak menurunkan over time
  - Dispancy latency yang rendah :
    - Penyisipan point preemsi sepanjang waktu system calls
    - Membuat keseluruhan kernel preemptable